```
Changes to the Distributed Checksum Clearinghouse source

    2001/12/19 18:55:53 Rhyolite Software DCC 1.0.40-1.49 $Revision$

1.0.40
    fix for syntax error in /var/dcc/libexec/cron-dccd from Dave Lugo
    deal with slow dccd response to dbclean

1.0.39
    keep dccd from going crazy with a crazy value for -q

1.0.38
    fix dccd core dump with Dave Lugo's help.
    improve dccd host name resolving helper process.
    improve misc/na-spam.
    `cdcc 'stats clear'` now also clears the list of clients seen by dccd.
    add a path of server-IDs to flooded checksum reports.
    increase the number of checksums recognized by the server.
    fix pthread error on SunOS and possibly AIX.
    use absolute path for `cdcc` in /var/dcc/libexec/stop-dccd as suggested
        by Sam Leffler.
    improve fuzzy ignoring of MIME multipart boundaries.


1.0.37
    deal with lack of -s in SunOS `logger`.
    dccd now has a helper process to wait for slow DNS servers to resolve
        the names of flooding peers.
    Deleting and restarting the DCC server's database now causes dccd to
        ask peers to re-flood their checksums.  This new feature required
        changing the flooding protocol.  DCC servers using the new protocol
        talk to servers using the old protocol after the old servers start
        their streams or with an explicit tag in the /var/dcc/flod file.
    `sendmail -bs` is used by some mail user agents such as pine.  In such
        cases the sendmail milter interface gives filters such as dccm a null
        pointer to what should be an IP address and a pointer to the
        string "localhost".  Dccm now acts as if such mail arrived from
        IP address 127.1.  This makes the common white list entry
        "ok IP localhost" effective for such mail.  Note that dccm deletes
        X-DCC header lines with its own brand from white listed messages,
        because they would otherwise be wrong and a potential vulnerability
        to bad guys.
    Fix `dccproc -o ofile` to include the X-DCC header in ofile.  If this fix
        is a problem, see `dccproc -C`
    add /var/dcc/libexec/na-spam and ng-spam to gather spam from
        news.admin.net-abuse.sightings
    fix start-dccd, start-dccm, and cron-dccd to support multiple dccd
        daemons in separate home directories.

1.0.36
    support for OSF1.
    handle msync() with only two parameters in old BSD/OS.
    try to fix rare core-dump in dccm whitelist parsing.
    fix error in misc/dccdnsbl.m4 noted by Michael Ghens.
    fix autoconf errors for SunOS noted by Sam Leffler
    add "log-del" option to /var/dcc/flod file
    fix recent damage to DCC{D,M}_ARGS in start_dcc{d,m}

1.0.35
    add DCC_LOG_FACILITY to dcc_conf as suggested by Sam Leffler.
```

```
	You must install the new homedir/dcc_conf with your parameters
	    to use it.
	fix recently introduced bug that kept dccd from automatically
	    running dbclean to expand the database.
	document the output of the dblist program in its man page.
	`configure --with-rundir` can be used to override the use of /var/run/dcc
	    for the PIDs of DCC daemons, sockets, and so forth.
	`configure --with-uid=dcc` creates Makefiles and scripts to install
	    and start DCC programs as the user "dcc"


1.0.34
	support for IRIX
	fix bug in setting libexecdir for configure
	change `cdcc stats` to show cumulative report counts
	increase maximum number of flooding peers from 16 to 32
	    and make it a compile-time parameter
	change $UID in misc/start-dcc{d,m} to the avoid reserved variable in
	    RedHat 6.2 as suggested by Michael Ghens
	fix bug in cron-dccd found by Michael Ghens and Dave Lugo
	remove mechanism for configuring the DCC home directory by setting
	    an environment variable before invoking `make`
	change the default value of the -u anon-delay parameter for dccd to 0.
	add "flood list" operation to `cdcc`
	look for sendmail for dccm in a FreeBSD "ports" package

1.0.33
	support for HPUX thanks to Richard Rauenzahn.
	check against "$USER" instead of "root" in start-dccm and start-dccd
	    as suggested by Luke Crawford.
	make the server rate limits configurable at compile-time.

1.0.32
	fix bug in local white lists that ignored changes in the count field
	by default, start-dccm no longer tells dccm to reject based on
	    message-ID checksums
	fix recently introduced bug that kept flooding off after the
	    hash table needs to be expanded.

1.0.31
	add SOCKS support.
	dccproc only logs errors unless given -d.  This should fix problems
	    in some mail systems using dccproc caused by network problems.
	fix permissions bugs related to using private map files
	the dcc_notspam sendmail macro used by dccm with -o must be non-empty
	    to be considered "set".

1.0.30
	fix man page installation on OpenBSD.
	fix bug in starting incoming floods on systems with IPv6 interfaces
	    but without what DCC recognizes as IPv6 support such as OpenBSD.
	deal with systems such as OpenBSD with lame mmap() support.
	speed up recognition of changes in the /var/dcc/flod file.
	use DCCM_REJECT_AT in /var/dcc/dcc_conf to also set the default
	    flooding threshold used by dccd when it is started by
	    /var/dcc/libexec/start-dccd
	add configure switches to not build dccm and the server
	`dccd -u` turns off `cdcc stats` from anonymous systems to avoid telling
	    strangers how many mail messages a small DCC server has seen.
```

```
1.0.29
    fix start-dccd to deal better with non-standard DCC home directories.
    dccproc is now like dccm and treats a missing Message-ID header.
        as if it were present and with a null value.
    do the right thing for DCC servers running on platforms where
        gethostname() fails completely on a short buffer instead of
        giving a prefix of the hostname.
    detect and quit on null hostname from gethostname().

1.0.28
    improve the handling of an already running daemon in by misc/start-dccd
    support mapping of ranges of server IDs when flooding reports
    yet more changes to deal with quoted-printable.  These changes
        generally cause the fuz1 checksum to differ.
    remove need for FEATURE(delay_checks) when reporting sendmail access_db
        hits to DCC server
    change body checksum to ignore '>' in "\n>From" because the '>' is
        often added for old UNIX MUAs.
    improve response of dccproc to 20KByte or larger To: headers.
    make `cdcc "file map2; load map2.txt"` act the same as
        `printf "file map2\nload map2.txt" | cdcc`
    dccm now treats a missing Message-ID header as if it existed but with
        a null value.

1.0.27
    change example scripts to deal with `expr` exiting with 1 and stopping
        them on Solaris
    fix client IDs larger than 65535
    detect and complain server IDs offered to `cdcc` as client IDs

1.0.26
    if dccm is already installed, try to build it even if the sendmail
        milter library is not available to prevent silent failures to
        install new versions of dccm.

1.0.25
    fix confusion if a quoted-printable sequence overlaps a buffer boundary.
    do not give up on remote servers if a local server responds with
        an ICMP unreachable error.

1.0.24
    minimize interpreting '=' in a URL as quoted-printable to make dccproc
        and dccm compute the same fuzzy checksums more often.

1.0.23
    fix confusion in dccproc about whether an initial line of a message
        that starts with blanks is a continuation of the last header line

1.0.22
    fix infinite loop and packet spew from dccproc when the clock jumps
    backward or jumps forward more than 1000 seconds.
    fix syslog process name on Solaris and AIX
    `dccproc -R` picks IP address out of standard Received: lines
    fix bugs in decoding quoted printable with broken soft ends of lines

1.0.21
    repair DCC server whitelist broken in 1.0.20

1.0.20
    support for Solaris
    describe ways to connect spam traps to the DCC in INSTALL.html
    move parameters from start-dccd, start-dccm, and cron-dccd to a common file
    add misc/rcDCC start-up script for Solaris and Linux
    fix byte-order bug in flood header server ID which requires changing
        the flood protocol.  To flood to version 1.0.19 or older versions
        of dccd, specifiy version 4 in the flod file line.
    removed locking file /var/dcc/map.lock
    change handling of spam sent simultaneously to white-listed and unlisted
        targets.  See the discussion of the new "REJECT_ONLY" action in the
        dccm man page.

1.0.19
    improve `cdcc stats` flood formatting
    fix `cdcc "host domain.com; stats all"`
    change dccproc to use the value of the Return-Path: header for the
        envelope-From checksum if the header is present and -f is not used.
    fix `dbclean -S -N` when the whitelist is empty
    add rough support for NetBSD.
    mention dccd in the INSTALL file.
    fix for parsing "-L error,LOCAL1.ERR" from Vincent Schonau

1.0.18
    add "clients -n" to cdcc
    add -C to dccproc

1.0.17
    add dccsight

1.0.16
    try again to deal with getifaddrs() without freeifaddrs().
    fix bug introduced in 1.0.15 that causes dccproc to require
        a white-list
    fix corruption of /var/dcc/map when dccproc is run with stderr not
        open and when the DCC server first fails to answer.

1.0.15
    make the sendmail {dcc_isspam} and {dcc_notspam} macros consistently
        override what dccm and the DCC server determine

1.0.14
    deal with systems that have getifaddrs() but not freeifaddrs().
    fix bogus response from server when a duplicate request from an
        anonymous client arrives before the original request has been
        scheduled to be answered.
    fix obscure double-trip bug in threaded client library.
    accept "rpt-ok" as well as "rpt_ok" in the ids file.
    fix /var/dcc/flod option scanning bug by dccd.
    'dccd -u 999999' turns off access by anonymous or unauthenticated clients.
    add -W to dccm to cause only explicitly listed targets to be protected
        by the DCC
    add a "reject" server-ID translation target in the flods file to
        not send or receive the reports of some servers.

1.0.13
    add RTT adjustment to cdcc load and add operations to allow a client
        to prefer servers despite worse RTT's

1.0.12
    in dccm count two intead of one open file for each active job against
        the system imposed limit on open files for automatically setting
        the value of -j for dccm and for automatically changing the soft
        resource limit.
    use the GNU autoconfig install script instead of `install -d` to create
        $(HOMEDIR)/libexec because GNU autoconfig does not detect install
        programs that do not understand -d
    rate limit complaints by dccd about unrecognized server IDs

1.0.11
    dccm tolerates null sender IP addres and hostname from `sendmail -bs`
        from sendmail 8.11.3 but perhaps not from 8.12.
    change -p for dccd and dbclean to -a to allow specification of entire
        server addresses.
    by default, dccd listens on separate UDP sockets so that clients receive
        responses from the same IP address to which they send requests.

1.0.10
    fix "bogus oflod complaint length 0" nonsense from server
    `cdcc stats` counts the clients seen in the last 24 hours, but
       `cdcc clients` displays all that fit in the cdcc buffer even if
        older than 24 hours
    the `configure` script looks at `make -v` to guess whether to generate
        gmake or make makefiles
    include list of common "dictionary attack" user names among the sample
        homedir files

1.0.9
    body checksums ignore effects of quoted-printable encoding
    deal with versions of gmake that do not understand ?=
    improve "clients" request of cdcc

1.0.8
    fix rate limiting bugs in the server
    fix local env-To whitelist

1.0.7
    fix locking bug when client whitelist file cannot be opened
    use `install -c` to not delete misc scripts
    fix server flood stalls when there are many stale or whitelisted
        reports

1.0.6
    fix bug in alternate dccm argv[0] in start-dccm
    fix bug in noticing changes to included white lists

1.0.5
    install cron-dccd, start-dccd, and start-dccm in $(HOMEDIR)/libexec

1.0.4
    fix server core-dump for repeated invalid admin. opcodes while
        tracing is enabled.
    add "clients" request to `cdcc`
    add "stats all" request to `cdc
    add homedir/start-dccm.sh
    /var/run/dccm.pid and /var/run/dccm depend on argv[0]
    white-lists can use "include pathname"
    dccm -o overrides -s
    dccm -o and -s have default values
    move /var/run/dccm and /var/run/dccm.pid to the directory /var/run/dcc
        and change the sendmail "feature" file misc/dcc.m4 to match
```

```
1.0.3
    improve flood ID mapping
    remove need to explicitly build before `make install`
```

dccm(8) DCC -- Distributed Checksum Clearinghouse dccm(8)

NAME

dccm - Distributed Checksum Clearinghouse Milter Interface

SYNOPSIS

```
dccm [-VdbxANQW] [-h homedir] [-p protocol:filename | protocol:port@host]
     [-m map] [-w whiteclnt] [-a IGNORE | REJECT | REJECT_ONLY]
     [-t type,[log-thold,][rej-thold]] [-g [not-]type] [-l logdir]
     [-r rejection-msg] [-s dcc_isspam] [-o dcc_notspam] [-j maxjobs]
     [-L ltype,facility.level]
```

DESCRIPTION

Dccm is a daemon built with the sendmail milter interface intended to connect sendmail to DCC servers. When built with the milter filter machinery and configured to talk to **dccm** in sendmail.cf, sendmail passes all email to **dccm** which in turn reports related checksums to the nearest DCC server. DCCM then adds an *X-DCC* SMTP header line to the message and optionally tells sendmail to reject it.

dccm sends reports of checksums related to mail received by DCC clients and queries about the total number of reports of particular checksums. A DCC server receives *no* mail, address, headers, or other information, but only cryptographically secure checksums of such information. A DCC server cannot determine the text or other information that corresponds to the checksums it receives. Its only acts as a clearinghouse of counts for checksums computed by clients. For complete privacy as far as the DCC is concerned, the checksums of purely internal mail or other mail that is known to not be unsolicited bulk can be listed in a white list to not be reported to the DCC server.

Since the checksums of messages that are white listed locally by **-W** or by **-w** *file* are not reported to the DCC server, **dccm** knows nothing about the total recipient counts for their checksums and so cannot add an *X-DCC* header line. Sendmail does not tell **dccm** about messages that are not received by sendmail via SMTP, including messags submitted locally and received via UUCP, and so they also do not receive *X-DCC* header lines.

The list of servers that **dccm** contacts is in a memory mapped file shared by local DCC clients. The file is maintained with cdcc(8). It is useful to put DCC parameters into the dcc_conf file and to start **dccm** with the start-dccm script.

When sendmail is not used, then **dccm** is not useful. dccproc(8) can often be used instead.

OPTIONS

The following options are available:

-V displays the version of the DCC Milter interface.

-d enables debugging output from the DCC client library. Additional **-d** options increase the number of messages. It is handy to use **-d** when **dccm** is first installed to detect and diagnose problems.

-b causes the daemon to not detach itself from the controlling tty and put itself into the background.

-x causes the daemon to try "extra hard" to contact a DCC server. Since it is usually more important to deliver mail than to report its checksums, **dccm** normally does not delay too long while trying to contact a DCC server. It will not try again for several seconds after a failure. With **-x**, unresponsive DCC servers cause mail to be temporarily rejected with RFC 821 400-series errors.

-A adds to existing DCC headers in the message instead of replacing existing headers for the brand of the current server.

-N neither adds, deletes, nor replaces existing DCC headers in the message. Each message is logged, rejected, and otherwise handled the same.

-Q only queries the DCC server about the checksums of messages instead of reporting and then querying. This is useful when **dccm** is used to filter mail that has already been reported to a DCC server by another DCC client. This can also be useful when applying a private white or black list to mail that has already been reported to a DCC server. No single mail message should be reported to a DCC server more than once per recipient, because each report will increase the apparent "bulkness" of the message.

-W is used to make recipient addresses white listed by default, to make it easier to manage systems where only a minority of users want unsolicited bulk mail to be rejected or discarded. This kludge of a feature is useful when addresses should by default not be protected against spam with the DCC. It causes mail sent only to *env_To* values or target addresses that are not listed in the **-w** *whiteclnt* file to be treated as if **-a** *IGNORE* were used. That means that the DCC server is queried and the *X-DCC* SMTP header is added but the message is delivered. Target addresses listed with "OK2" in the *whiteclnt* file are treated neutrally or as if they were not listed and with the **-W** option not specified.

Mail addressed simultaneously to unlisted targets and targets listed with "OK2" is treated as if **-W** were not used. Otherwise, "OK2" is ignored for *env_To* values.

Note that all valid forms of a target address are treated independently. For example, if user@host.domain.com and user@domain.com are delivered to the same mailbox and should be treated the same, then both must be listed (or not) in the *whiteclnt* file.

See also the discussion of **-w**.

-h *homedir*
overrides the default DCC home directory, which is often /var/dcc.

-p *protocol:filename* | *protocol:port@host*
specifies the protocol and address by which sendmail will contact **dccm**. The default is a UNIX domain socket in the "run" directory, often */var/run/dcc/dccm*. (See also **-R**) This protocol and address must match the value in *sendmail.cf*. This mechanism can be used to connect dccm on one computer to sendmail on another computer when a port and host name or IP address are used.

-m *map*
specifies a name or path of the memory mapped parameter file instead of the default *map* in the DCC home directory. It should be created with the **cdcc(8)** command.

-w *whiteclnt*
specifies an optional file containing SMTP client IP addresses, SMTP envelope values, and header values of mail that is not spam, does not need a DCC header, and whose checksums should not be reported to the DCC server. It can also contain checksums marking spam. Local whitelist *env_To* values are handy for exempting destination addresses such as Postmaster from filtering or for marking addresses that should never receive mail.

If the pathname is not absolute, it is relative to the DCC home directory. The format of the **dccm** whiteclnt file is the same as the *whitelist* file required by **dbclean(8)** and the optional *whiteclnt* file used by **dccproc(8)**. See **dcc(8)** for a description of DCC white and blacklists. DCC server *whitelist* files are required while this *whiteclnt* file is optional. Because the contents of the *whiteclnt* file are used frequently, a companion file is automatically created and maintained. It has the same pathname but with an added suffix of *.dccw*. It contains a memory mapped hash table of the main file.

A local white listing entry ("OK) or two or more semi-white listings ("OK2") for one of the message's checksums prevents all of the message's checksums from being reported to the DCC server and the addition of a *X-DCC* header line by **dccm**. A local white listing entry for a checksum also prevents rejecting or discarding the message based on DCC recipient counts as controlled by **-a** and **-t**. Otherwise, one or more checksums with blacklisting entries ("MANY") cause all of the message's checksums to be reported to the server with an addressee count of "MANY".

If the message has a single recipient, an *env_To* local *whiteclnt* entry of "OK" for the checksum of its recipient address acts like any other *whiteclnt* entry of "OK." When the SMTP message has more than one recipient, the effects can be complicated. When a message has several recipients with some but not all listed in the *whiteclnt* file, **dccm** tries comply with the wishes of the users who want filtering as well as those who don't by silently not delivering the message to those who want filtering (i.e. are not white-listed) and delivering the message to don't want filtering. If there are too many users who want filtering (about 1024 total bytes of addresses), then the wishes of those who do not want filtering are ignored and the message is rejected or discarded for all recipients. The **-A** *REJECT_ONLY* option effectively reduces the limit of "too many users" to 1.

Consider the **-W** option for implicitly or by default white-listing *env_to* values.

-a *IGNORE* | *REJECT* | *REJECT_ONLY* | *DISCARD*
specifies the action taken when DCC server counts or **-t** thresholds say that a message is unsolicited bulk. *IGNORE* causes the message to be unaffected except for adding a header line to the message. Spam can also be *REJECT*ed, or accepted and silently *DISCARD*ed without being delivered to local mailboxes.

With an action of *REJECT* or *DISCARD*, spam sent to both white-listed targets and non-white-listed targets is delivered to white-listed targets and if possible, silently discarded for non-white-listed targets. This is not possible if there are too many non-white-listed targets to be saved in a buffer of about 500 bytes.

An action value of *REJECT_ONLY* causes spam addressed to both non-white-listed and white-listed targets to treated as if there were too many be targets and so always rejected for all targets. If there are too many non-whitelisted targets, the preference of the white-listed to receive the message is ignored and it is rejected.

The default is *REJECT*.

Determinations that mail is or is not spam from sendmail via **-s** or **-o** completely override **-a**.

-t *type,*[*log-thold,*][*rej-thold*]
sets logging and "spam" thresholds for checksum *type*. The checksum types are *IP*, *env_From*, *From*, *Subject*, *Message-ID*, *Received*, *body*, and *fuz1*. The special type *ALL* sets thresholds for all types, but is unlikely to be useful except for setting logging thresholds. Rej-thold and log-thold are numbers, the null string for infinity, or the string *MANY* indicating millions of targets. Counts from the DCC server as large as the threshold for any single type are taken as sufficient evidence that the message should be logged or rejected. *Log-thold* is the threshold at which messages are logged. It can be handy to log messages at a lower threshold to find solicited bulk mail sources such as mailing lists. The logging threshold cannot be higher than the rejection threshold.

The default is infinity, so that nothing is discarded or logged. A common choice is
-t *body,50,100* **-t** *fuz1,50,100* **-t** *received,50,100*
to reject or discard, as controlled by **-a**, mail with common bodies or Received headers except as overridden by the white list of the DCC server and local **-o**, **-g**, **-n**, **-n**, and **-w**.

-g [*not-*]*type*
indicates that white-listed or *OK* or *OK2* "counts" from the DCC for a type of checksum are to be believed. They should be ignored if prefixed with *not-*. *Type* is one of the same set of strings as for **-t**. Only *IP*, *env_From*, and *From* are likely choices. By default all three are honored, and hence the need for *not-*.

-l *logdir*
specifies a directory in which copies of messages processed by **dccm** are kept. See the FILES section below.

-R *rundir*

specifies the "run" directory where the UNIX domain socket and file containing the daemon's process ID are stored. The default value is often /var/run/dcc.

-r *rejection-msg*

specifies the rejection message for unsolicited bulk mail. It replaces the default, "5.7.1 550 mail from X rejected by BRAND DCC". A common alternate is "4.7.1 451 Access denied by DCC" to tell the sender to continue trying. If the message does not start with a recognized error type and number, type 5.7.1 and number 550 are used.

-s *dcc_isspam*

specifies the name of a sendmail macro which, when set by sendmail.cf rules, causes a message to be reported to the DCC server as having been addressed to "MANY" recipients. This macro must also be added to the Milter.macros.envrcpt option statement in sendmail.cf as in the example "Feature" file dcc.m4. The default macro name is *dcc_isspam* or *{dcc_isspam}*. If the *dcc_notspam* macro specified with **-o** is set to a non-empty string by sendmail, then the *dcc_isspam* macro is ignored.

If the value of the *dcc_isspam* is null, **dccm** uses its default smtp rejection messages, as controlled by **-a** and **-r**. If the value of the *dcc_isspam* macro starts with "DISCARD", the mail message is silently discarded instead of rejected. This can be handy for keeping "spammers" from knowing they are sending to "spam traps." If value of the macro not null and does not start with "DISCARD", it is used as the SMTP error message given to the SMTP client trying to send the rejected message. The message starts with an optional SMTP error type and number followed by text. The **-a** option does not effect the disposition of the message sendmail has declared is spam.

When the *dcc_isspam* is set, the message is rejected or discarded despite local or DCC database white-list entries. The local white-liste does control whether the message's checksums will be reported

to the DCC server and an *X-DCC* SMTP header line will be added.

-o *dcc_notspam*

specifies the name of a sendmail macro which, when set by sendmail.cf rules, causes a message not be considered unsolicited bulk despite any evidence to the contrary. It also prevents **dccm** from reporting the checksums of the message to the DCC server and from adding a header line. This macro name must also be added to the Milter.macros.envrcpt option statement in sendmail.cf as in the example "Feature" file dcc.m4. The default macro name is *dcc_notspam*.

When set, *dcc_notspam* overrides DCC threshlds that say the message should be rejected as well as the effects of the *dcc_isspam* macro specified with **-s**

-j *maxjobs*

limits the number of simultaneous requests from sendmail that will be processed.

-L *ltype,facility.level*

specifies how messages should be logged. *Ltype* must be *error* or *info* to indicate which of the two types of messages are being controlled. *Level* must be a **syslog(3)** level among *EMERG, ALERT, CRIT, ERR, WARNING, NOTICE, INFO,* and *DEBUG*. *Facility* must be among *AUTH, AUTHPRIV, CRON, DAEMON, FTP, KERN, LPR, MAIL, NEWS USER, UUCP,* and *LOCAL0* through *LOCAL7*. The default is equivalent to

-L *info,MAIL.NOTICE* **-L** *error,MAIL.ERR*

FILES

/var/dcc is the DCC home directory in which other files are found.

libexec/start-dccm
is a script often used to the daemon.

dcc/dcc_conf
contains parameters used by the scripts to start DCC daemons and cron jobs.

logdir is an optional directory containing marked mail. Each file in the directory contains one message, at least one of whose checksums exceeded its or that is interesting for some other reason. Each file starts with lines containing the date when the message was received, the IP address of the SMTP client, and SMTP envelope values. Those lines are followed by the body of the SMTP message including its header as it was received by sendmail and without any new or changed header lines. Only approximately the first 32 KBytes of the body are recorded. The checksums for the message follow the body. They are followed by lines indicating that the **-t** *log-thresh-* or **-o** *dcc_notspam* macros were set by sendmail or one of the checksums is white- or blacklisted by the **-w** *whiteclnt* file. Each file ends with the *X-DCC* header line added to the message and the disposition of the message including SMTP status message if appropriate.

map is the memory mapped file of information concerning DCC servers in the DCC home directory.

whiteclnt contains the client white list in the format described in **dcc(8)**.

whiteclnt.dccw
is a memory mapped hash table of the *whiteclnt* file.

/var/run/dcc/dccm.pid
contains daemon's process ID. The string ``dccm'' is replaced by the file name containing the daemon to facilitate running multiple daemons, probably connected to remote instances of sendmail using TCP/IP instead of a UNIX domain socket. See also **-R**.

/var/run/dcc/dccm
is the default UNIX domain socket used by the sendmail milter interface. See also **-R**.

sendmail.cf
is the **sendmail(8)** control file.

EXAMPLES

Dccm should be started before sendmail with something like the script libexec/start-dccm. It looks for common DCC parameters in the file dcc_conf in the DCC home directory.

Those numbers should modified to fit local conditions. It might be wise to replace the "100" numbers with much larger values or with "MANY" until a few weeks of monitoring the log directory show that sources of mailing lists are in the server's whitelist file (see **dccd(8)**) or the local *whiteclnt* file.

It is usually necessary to regularly delete old log files with a script like libexec/cron-dccd.

Sendmail must be built with the milter interface, such as by creating a *devtools/Site/site.config.m4* or similar file containing something like the following lines:

```
APPENDDEF(`conf_sendmail_ENVDEF', `-D_FFR_MILTER=1')
APPENDDEF(`conf_libmilter_ENVDEF', `-D_FFR_MILTER=1')
```

Appropriate lines invoking the milter interface must be added to sendmail.cf. It should be sufficient to copy the dcc.m4 file to the sendmail 8.11 cf/feature directory and add the line

```
FEATURE(dcc)
```

to the local .mc file.

SEE ALSO

cdcc(8), **dbclean(8)**, **dcc(8)**, **dccd(8)**, **dblist(8)**, **dccproc(8)**, **dccsight(8)**, **sendmail(8)**.

HISTORY

Implementation of **dccm** was started at Rhyolite Software in 2000. This describes version 1.0.40.

BUGS

On many systems with sendmail 8.11.3 and preceding, a bug in the sendmail milter mechanism causes **dccm** to die with a core file when given a signal.

Systems without **setrlimit(2)** and **getrlimit(2)** can have problems with the default limit on the number of simultaneous jobs, the value of **-j**. Every job requires two open files. One is the socket connected to sendmail and the other is the **dccm** log file. These problems are usually seen with errors messages that say something like

```
dccm[24448]: DCC: accept() returned invalid socket
```

A fix is to use a smaller value for **-j** or to allow **dccm** to open more files.

Man(1) output converted with man2html modified for the DCC $Date 2001/04/29 03:22:18 $

dccd(8) DCC -- Distributed Checksum Clearinghouse **dccd(8)**

NAME

dccd - Distributed Checksum Clearinghouse Daemon

SYNOPSIS

dccd [**-VbQ**] **-i** *server-ID* **-n** *brand* [**-h** *homedir*]
[**-a** [*server-addr*][*,server-port*]] [**-I** *host-ID*] [**-q** *qsize*]
[**-t** [*type*],threshold] [**-T** *tracemode*] [**-u** *anon-delay*] [**-C** *dbclean*]
[**-L** *ltype,facility.level*]

DESCRIPTION

Dccd receives reports of checksums related to mail received by DCC clients and queries about the total number of reports of particular checksums. A DCC server never receives mail, address, headers, or other information from clients, but only cryptographically secure checksums of such information. A DCC server cannot determine the text or other information that corresponds to the checksums it receives. It only acts as a clearinghouse of total counts of checksums computed by clients.

Each DCC server or close cluster of DCC servers is identified by a numeric *server-ID*. Each DCC client is identified by a *client-ID*, either explicitly listed in the *ids* file or the special anonymous client ID. Many computers are expected to share a single *client-ID*. A *server-ID* is less than 32768 while a *client-ID* is between 32768 and 16777215. DCC server IDs need be known only to DCC servers and the people running them. The passwords associated with DCC server IDs should be protected, because DCC servers listen to commands authenticated with server IDs and their associated passwords. Each client that does not use the anonymous ID must know the client-ID and password used by each of its servers. A single client computer can use different passwords with different server computers. See the *ids* file.

A white list of known good (or bad) sources of email prevents legitimate mailing lists from being seen as unsolicited bulk email by DCC clients. The white list should include at least the IP address 127.1. The white list used by a DCC server is built into the database when old entries are moved. See **dbclean(8)**. DCC servers exchanging (or "flooding") reports should have a common white list. Many failures to include white list entries are detected and counted as "white" by the **stats** server command by **cdcc(8)**. Each DCC client has its own, local white list.

The effectiveness of a Distributed Checksum Clearinghouse increases as the number of subscribers increases. Flooding reports of checksums among DCC servers increases the effective number of subscribers to each server. Each **dccd** daemon tries to maintain TCP/IP connections to the other servers listed in the *flod* file, and send them reports containing checksums with total counts exceeding thresholds. Changes in the *flod* file are noticed automatically within minutes.

Controls on report flooding are specified in the *flod* file. Each line specifies a hostname and port number to which reports should be flooded, a server ID to identify and authenticate the output stream, a server ID to identify and authenticate an input stream from the same server, and flags with each ID. The ability to delete reports of checksums is handy, but could be abused. If *no-del* is present among the *in-opts* options for the incoming ID, incoming delete requests are logged and then ignored. Floods from DCC "brands" that count only mail to "spam traps" and whose servers use the **-Q** option to count extremely "bulk" mail should be marked with *traps*. They can be seen as counting millions of targets, so the *traps* flag on their *flod* file entry changes their incoming flooded reports counts to "many."

Dccd automatically checks its *flod* and *ids* files periodically. **Cdcc(8)** has the server commands **new ids** and **flood check** to tell **dccd** to check those two files immediately. Both files are also checked for changes in response to the SIGHUP signal.

OPTIONS

The following options are available:

-V displays the version of the DCC server daemon.

-b causes the server to not detach itself from the controlling tty or put itself into the background.

-Q causes the server to treat reports of checksums as queries except from DCC clients marked trusted in the *ids* file with *rpt-ok*. See **-u** to turn off access by anonymous or unauthenticated clients

-i *server-ID*
specifies the ID of this DCC server. Each server identifies itself as responsible for checksums that it forwards to other servers.

-n *brand*
is an arbitrary string of letters and numbers that identifies the organization running the DCC server. The brand is required, and appears in the SMTP *X-DCC* headers generated by the DCC.

-h *homedir*
overrides the default DCC home directory, which is often */var/dcc*.

-a [*server-addr*][*,server-port*]
adds an hostname or IP address to the list of local IP addresses that the server answers. Multiple **-a** options can be used to specify a subset of the available network interfaces or to use more than one port number. The default is to listen on all local IP addresses. It can be useful to list some or all of the IP addresses of multi-homed hosts to deal with local or remote firewalls. *Server-port* is 6277 by default and is the UDP port at which DCC requests are received and the TCP port for incoming floods of reports.

If *server-addr* is absent and if the **getifaddrs(8)** function is supported, separate UDP sockets are bound to each configured network interface so that each DCC clients receives replies from the IP addresses to which corresponding request are sent. If **dccd** is started before all network interfaces are turned on or there are interfaces that are turned on and off or change their addresses such as PPP interfaces, then the special string @ should be used to tell **dccd** to bind to an IN_ADDRANY UDP socket.

Outgoing TCP connections to flood checksum reports to other DCC servers used the IP address of a single **-a** option, but only if there is single option. See also the *flod* file.

-I *host-ID*

changes the server's globally unique identity from the default value consisting of the first 16 characters of the host name. *Host-ID* is a string of up to 16 characters to be used instead of the first 16 characters of the system's hostname.

-q *qsize*

specifies the maximum size of the queue of requests from anonymous or unauthenticated clients.

-t [*type*],*threshold*

sets the threshold below which checksum reports are not sent or flooded to peer DCC servers. Checksums whose total counts are less than to the number *threshold* are not sent. If *threshold* is the string "many," a value of millions is understood. If *type* is absent, the *threshold* replaces the default values for the body checksums.

This mechanism allows cooperating DCC servers to share only the checksums of bulk mail and significantly reduces inter-server communications. The thresholds should be larger than the number of addressees of typical private email but not much larger, because reports of checksums that total less than their thresholds can be flooded as many extra times as there are other thresholds. By default, the thresholds for the body checksums, *body* and *fuz1*, are 20. The thresholds for the other checksums are so high by default that by themselves they can never cause reports to be flooded.

Reports containing any checksums marked "OK or "OK2" are not sent to other servers. This reduces the bandwidth needed for the inter-server flooding, the sizes of DCC database files, and helps protect the privacy of email of clients of a DCC server.

-T *tracemode*

causes the server to trace or record some operations. *tracemode* must be one of the following:

ALL turns on all tracing.
ADMN traces administrative requests from the control program. See cdcc(8).
ANON traces errors by anonymous clients.
CLNT traces errors by authenticated clients.
RLIM notes rate-limited messages.
QUERY traces all queries and reports.
RIDC produces some messages concerning the report-ID cache that is used to detect duplicate reports from clients.
FLOOD enables messages about inter-server flooding.

-u *anon-delay*

changes the number of milliseconds anonymous or unauthenticated clients must wait for answers to their queries and reports. The purpose of this delay is to discourage the use by strangers of a DCC server. The default value is 0. A value of 999999 or the string *forever* turns off all anonymous or unauthenticated access not only for checksum queries and reports but also **cdcc(8) stats** requests.

-C *dbclean*

changes the default name or path of the program used to rebuild the hash table when it becomes too full. The default value is *libexec/dbclean* in the DCC home directory.

-L *ltype,facility.level*

specifies how messages should be logged. *Ltype* must be *error* or *info* to indicate which of the two types of messages are being controlled. *Level* must be a **syslog(3)** level among *EMERG, ALERT, CRIT, ERR, WARNING, NOTICE, INFO,* and *DEBUG*. *Facility* must be among *AUTH, AUTHPRIV, CRON, DAEMON, FTP, KERN, LPR, MAIL, NEWS, USER, UUCP,* and *LOCAL0* through *LOCAL7*. The default is equivalent to

```
-L info,MAIL.NOTICE -L error,MAIL.ERR
```

FILES

/var/dcc is the DCC home directory containing data and control files.

dcc_db is the database of checksums.

dcc_db.hash is the database hash table.

flod contains lines controlling DCC flooding of the form:

host[*,port*] [*;src*] *rem-ID* [*passwd-ID*] [*o-opts*] [*i-opts*] [*vers*]

where absent optional values are signaled with "-" and

host is the IP address or name of a DCC server.

port is the name or number of the UDP port used by the server.

src is the IP address or host name from which the outgoing connection should come.

rem-id is the server-ID of the remote DCC server.

passwd-ID is a possibly phony ID whose first password is used to sign checksum reports sent to the remote system and either of whose passwords must be used to sign incoming reports. If it is absent or "-", outgoing floods are signed with the first password of the local server in the ids file and incoming floods must be signed with either password of the remote server-ID.

i-opts and *o-opts* are comma-separated lists of

- *off* turns off flooding to the remote or local system.
- *traps* indicates that the remote sending or local receiving system has only "spam traps."
- *no-del* says delete requests are refused by the remote or local system.
- *log-del* logs incoming delete requests to generate messages.
- *passive* is used to tell a server outside a firewall to expect a peer inside and using the SOCKS protocol to create both of the pair of input and output TCP connections used for flooding. The peer inside the firewall should use *SOCKS* on its flod file entry for this system.
- *SOCKS* is used to tell a server inside a firewall that it

should create both of the TCP connections used for flooding and that SOCKS protocol should be used. The peer outside the firewall should use *passive* on its flod file entry for this system.

ID1->ID2 converts *ID1* in flooded reports to *ID2*. Either *ID1* or *ID2* may be the string *self* to specify the server's own ID. *ID1* can be the string *all* to specify all IDs or a pair of IDs separated by a dash to specify a range. *ID2* can be the string *reject* to not send or refuse incoming checksums. Only the first matching conversion is applied. For example, when more than conversion matches a given report, such as *self->self,all->reject,* the first conversion is applied and the others are ignored.

vers specifies the version of the DCC flooding protocol used by the remote DCC server with a string such as *version2*.

Removing a line in the flod file, using the **cdcc(8)** command to tell the server **flood check**, and then restoring the line will cause the server to retransmit all eligible reports to the named remote server.

flod.map is an automatically generated file in which **dccd** records its progress sending or flooding reports to DCC peers.

ids contains the IDs and passwords known by the DCC server. An ids file that can be read by others cannot be used. It contains blank lines, comments starting with "#" and lines of the form:

id[,rpt-ok] *passwd1* [*passwd2*]

where

id is a DCC *client-ID* or *server-ID*.

Rpt-ok if present overrides the **-Q** argument by saying that this particular client is trusted to report only checksums for unsolicited bulk mail or spam.

passwd1 is the password currently used by clients with identifier *id*. It is a 1 to 32 character string that does not contain blank, tab, newline or carriage return characters.

passwd2 is the optional next password that those clients will use. A DCC server accepts either password if both are present in the file.

whitelist contains the server white list. It is not used directly but is loaded into the database when **dbclean(8)** is run.

EXAMPLES

dccd is usually started with other system daemons with something like the script misc/start-dccd. It uses values in the file dcc_conf in the DCC home directory to start the server.

The following is useful for cleanly stopping the daemon:

```
cdcc 'id 100; stop'
```

Again, the ID of the local server must be used instead of "100."

Unless old reports are removed from the database, it grows too large. **dbclean(8)** should be run daily with script like /var/dcc/libexec/cron-dccd.

SEE ALSO

cdcc(8), **dcc(8)**, **dbclean(8)**, **dblist(8)**, **dccm(8)**, **dccproc(8)**.

HISTORY

dccd is based on an idea from Paul Vixie. It was designed and written at Rhyolite Software starting in 2000. This document describes version 1.0.40.

December 19, 2001 5

Man(1) output converted with man2html modified for the DCC $Date 2001/04/29 03:22:18 $

NAME

cdcc - Control Distributed Checksum Clearinghouse

SYNOPSIS

cdcc [**-Vd**] [**-h** *homedir*] [**-c** *ids*] [*op1 op2* ... [-]]

DESCRIPTION

Cdcc is used to clear, control, and query the control file used by Distributed Checksum Clearinghouse clients such as dccm(8). The hostnames, UDP port numbers, IDs, and passwords local clients use to talk to servers as well as IP addresses, round trip times, and other information are contained in the *map* file. While **cdcc** is set-UID, it uses the real UID only when accessing the *map* file. It refuses to display sensitive information such as passwords unless the real UID is the same as the effective UID. Note that **cdcc** needs to be set to a UID that can read and write the *map* file, but that UID need not be 0.

Cdcc is also used to send commands to DCC servers to tell them to stop, reload their lists of DCC IDs, turn on tracing, and so forth.

Many commands sent to DCC servers require a numeric DCC ID and a password recognized by the server. A DCC password is a 1-32 character string that does not contain blank, tab, newline or carriage return characters. The ID is specified with the **id** operation. If **cdcc** is run with a real UID that can read the *ids* file and a password is not specified (see the **password** operation), then the current password for the specified ID in the *ids* file will be used. If no *ids* file is available and a password and DCC ID are not specified, **cdcc** uses the anonymous DCC client ID. DCC servers do not expect a password from clients using the anonymous client ID, but they also won't honor control requests.

Operations that modify the *map* file can only be performed when the real UID is sufficient to modify the file directly. Trying to perform an operation that requires a password without specifying a server ID or without using a UID that can access the *ids* file produces an error message complaining about a "privileged operation."

Commands and operations are read from the command line or from stdin. A series of *op1 op2* ... operations followed a - (a dash) causes operations to be read from stdin after the command line operations are processed. Semi-colons or newlines separate commands in UNIX command-line "words," as well as when commands are read from stdin. Since each command line operation must be a shell "word," quotes are often required as in

```
% cdcc "load map.txt"
```

or

```
% cdcc "host localhost;info" stats
```

OPTIONS

The following options are available:

-V displays the version of the DCC controller.

-d enables debugging output from the DCC client library. Additional **-d** options increase the number of messages.

-h *homedir*
overrides the default DCC home directory, which is often /var/dcc. See the **homedir** operation.

-c *ids*
specifies file containing DCC IDs and passwords known by the local DCC server. An *ids* file that can be read by others cannot be used. The format of the *ids* file is described in **dccd(8)**.

op1 op2 ...
are operations or commands such as "id 100; stop". Commands or operations specified on the command line are performed before the first interactive request. The last command can be - to specify that additional commands should be read from stdin.

OPERATIONS

Local operations include the following:

help [*command*]
lists information about one or all available commands and operations.

exit stops **cdcc**

homedir [*path*]
displays or specifies the DCC home directory.

file [*map*]
displays or specifies the name or path of the map file. The string "-" specifies the default file *map* in the DCC home directory.

new map [*map*]
creates a new, empty file for DCC server host names, port numbers, passwords, and so forth. There must not already be a file of the same name. The default is *map* in the DCC home directory.

delete *host*[,*port-number*]
deletes the entry in the *map* file for *host* and UDP *port-number.*

add *host*[,*port-number*] [*RTT+adj*|*RTT-adj*] [*client-ID* [password]]
adds an entry to the *map* file. The *port-number* can be "-" to specify the default DCC server port number.

An adjustment to the round trip time is a multiple of 10 milliseconds between -1270 and +1270 following the string *RTT*. The adjustment is added to the average measured round trip time when the DCC client software picks the "nearest" DCC server, or the server with the smallest RTT. If an IP address is mentioned more than once in the list of servers, for example because it is among the addresses for more than one server name, conflicts among RTT adjustments are resolved by picking the adjustment with the largest absolute value.

If both the client-ID and the password are absent, the anonymous client-ID, 1, is used. The string *anon* is equivalent to the anonymous client-ID. A null password string is assumed if the password is missing and the client-ID is 1 or also missing.

load *info-file*
loads the current parameter file with the hostnames, port numbers, IDs, and passwords in *info-file*. Standard input is understood if *info-file* is "-".

A suitable file can be created with the **info** operation and consists of blank lines and comment lines starting with '#', and of lines in the same format as the arguments to the **add** operation. Note that the output of the **info** command must saved by a user privilege to read the map file, or the information will lack passwords.

host [*hostname*]
specifies the hostname of the DCC server to which commands should be sent. If *hostname* is "-", the current default DCC server is

chosen.

port [*port-number*]
specifies the UDP port number of the DCC server to which commands should be sent.

password *secret*
specifies the password with which to sign commands sent to the DCC server specified with the **server** and **port** operations.

id [*ID*]
specifies or displays the numeric DCC ID for commands sent to the DCC server specified with the **server** and **port** operations. If no password is specified with the **password** command, the password is sought in the local *ids* file, if it exists.

info displays information about the connections to DCC servers. It starts with the current date and name of the current *map* file or says that **cdcc** is using the implicit file created with the **server** and **port** operations. It then says when hostnames will next be resolved into IP addresses, the smallest round trip time to the IP addresses of known DCC servers. The hostname, UDP port number (or dash if it is the default), DCC client ID, and password (if **cdcc** is used by a privileged user) are shown in one line per configured DCC server.

The currently preferred IP address is indicated by an asterisk. The "brand" of the server, its DCC ID, and its IP address are displayed in one line per IP address. The recent performance of the server at each IP address is displayed in a second line per address. The second line ends with the measured delay imposed by the server on requests with this client's ID.

RTT measures the round trip time to the DCC servers. It does this by discarding accumulated information and forcing a probe of all listed server IP addresses.

Beware that when run with sufficient privilege, the **RTT** operation is like the **info** and **load** operations and displays cleartext passwords.

debug [*on* | *off*]
enables or disables debugging information from the DCC client library.

IPv6 [*on* | *off*]
sets a switch to cause clients using the map file to try to use IPv6.

SOCKS [*on off*]
sets a switch to cause DCC clients using the map to use the SOCKS5 protocol, if they have been built with a SOCKS library. The socks library linked with the DCC client must be configured appropriately, often including knowing which DCC servers must be connected via the SOCKS proxy and which can be reached directly. DCC clients use SOCKS functions such as Rsendto() with all or no servers. Host name resolution should not be done by the SOCKS library. Use IP addresses for servers that cannot be resolved within a firewall.

DCC SERVER COMMANDS

Commands that can be sent to a DCC server include the following. All except the **stats** command must be used with the server's *ID* specified with the **id** command.

delck *type string*
asks the server to delete the *type* checksum corresponding to the *string*.

delck body *filename*
asks the server to delete the checksums for the message contained in *filename*. Stdin is understood if *filename* is "-". There must be no headers but only the body of the message in the file. Note that the message is assumed to not contain MIME multipart boundaries.

delck hex type hex1 hex2 hex3 hex4
asks the server to delete the *type* checksum with value *hex1 hex2 hex3 hex4*.

pck *type string*
prints the checksum corresponding to *string* in the format used by **dblist(8)**.

pck body *filename*
displays the checksums for the message contained in *filename*. Stdin is understood if *filename* is "-". There must be no headers but only the body of the message in the file. Note that the message is assumed to not contain MIME multipart boundaries.

stats
displays current status and statistics from the DCC server.

stats clear
displays current status and statistics from the DCC server and then clears them.

stats all
switches to the most recently specified map file and then obtains the statistics from all known servers just as the **stats** server command does.

clients
displays some of the most recently clients seen by the server. **clients -n** turns off the display of the names of clients.

stop
tells the DCC server to exit.

new IDs
tells the DCC server to reload its DCC *ids* file. This is handy to cause the server to notice changes in the file.

flood check
tells the DCC server to check for changes in the *flod* file and try to restart any of the streams to peers that are broken.

flood shutdown
tells the DCC server to cleanly stop flooding checksums to and from peers. The server will wait for sending and receiving peers to agree to stop.

flood halt
tells the DCC server to abruptly stop flooding checksums to and from peers.

flood rewind
tells the DCC server to ask its peers to rewind and resend the streams of checksums they are sending.

flood resume
tells the DCC server to turn on flooding if it is off, check the streams to peers, and try to restart them if necessary. This is

handy to speed up notice of changes to the server's *flod* file.

flood list
fetches the list of current incoming and outgoing floods.

DB unlock
is used by **dbclean** to tell the server that the database expiration has begun.

DB new
is used by **dbclean** to tell the server that the database cleaning is complete.

cdcc exits 0 on success, and >0 if an error occurs in operations specified on the command line.

FILES

/var/dcc DCC home directory

map
: memory mapped file in the home DCC home directory of server host names, port numbers, passwords, measured round trip times (RTT), and so forth.

ids
: list of IDs and passwords, as described in **dccd(8)**. It is only required by systems running the DCC server, but is used by **cdcc** if available.

SEE ALSO

dbclean(8), **dcc(8)**, **dccd(8)**, **dblist(8)**, **dccm(8)**, **dccproc(8)**, **dccsight(8)**.

HISTORY

Implementation of **cdcc** was started at Rhyolite Software in 2000. This describes version 1.0.40.

Man(1) output converted with man2html modified for the DCC $Date 2001/04/29 03:22:18 $